National College of Ireland

TB-DBN: A hybrid deep learning architecture with IBBA Optimization for enhanced phishing URL Detection

MSc Research Practicum
MSc in Artificial Intelligence

Sruthi Reddy Kavva
Student ID: 23314915

School of Computing
National College of Ireland

Supervisor: Kislay Raj

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sruthi Reddy Kavva
Student ID:	23314915
Programme:	MSc in Artificial Intelligence **Year:** 2024-2025
Module:	MSc Research Practicum
Supervisor:	Kislay Raj
Submission Due Date:	15/09/2025
Project Title:	TB-DBN: A hybrid deep learning architecture with IBBA Optimization for enhanced phishing URL Detection

Word Count: 7445 **Page Count:** 23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sruthi Reddy Kavva

Date: 14/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

TB-DBN: A hybrid deep learning architecture with IBBA Optimization for enhanced phishing URL Detection

Sruthi Reddy Kavva
23314915

Abstract

From 2022 to 2024, phishing increased by 61%, necessitating the use of increasingly sophisticated detection techniques beyond basic blacklists. This study offers a novel hybrid structure that combines Transformer-based contextual representation and DBN hierarchical learning, which is further improved by the Intelligence Binary Bat Algorithm (IBBA). The lack of adaptive preprocessing, the unexplored hybrid transformer-DBN architectures, and the absence of metaheuristic optimisation of deep hybrid systems are the three main shortcomings that the study addresses.

The TB-DBN system enables adaptive preprocessing, which adjusts the feature extraction based on the characteristics of the URLs. 169 features that fall into eight categories are extracted. Through the use of fusion, the architectural design integrates the DeBERTa-v3-base transformer and the multi-layer DBNs, and the IBBA regularly optimises the hyperparameters.

A 96.38% F1-score (±0.0023) is obtained from the test on 11,430 URLs, significantly outperforming DBNs (81.4%) and unified transformers (87.5%). The contribution made by this work goes beyond performance indicators; it provides a unique approach to combining languages and structure to create patterns that prevent millions of dollars in losses every year. Using the same methodology across all cybersecurity domains is made easier by an optimised design through IBBA and an adaptive preprocessing framework. One significant advancement in combating constantly evolving threats is the 8.5% relative improvement over single-model approaches. An important step forward for upcoming projects and broad industry adoption is open-source software. This study shows that the future of reliable phishing detection lies not in larger models but in intelligent architectural integration.

1 Introduction

Phishing attempts, aimed at obtaining sensitive information, constitute a continual and versatile threat to the security of digital communication. Recent data indicates that phishing attacks increased by 61% from 2022 to 2024, leading to global losses exceeding $10 billion yearly (Haq et al., 2024). Phishing attacks are becoming increasingly sophisticated. To evade detection by standard methods, the offenders are employing intricate modifications and sophisticated obfuscation in the URLs. The contemporary security landscape necessitates very sophisticated detection systems that remain effective despite evolving threat methodologies.

Contemporary techniques for identifying phishing URLs predominantly depend on blacklists and rule sets, which are ineffective for zero-day exploits and real-time phishing URLs. Machine learning techniques have demonstrated superior outcomes compared to traditional methods. Ferdaws and Majd (2024) achieve a performance rate of 98.1% using LSTM-based techniques; nonetheless, these systems encounter challenges related to feature engineering, adaptation to various attack vectors, and processing speeds. Contemporary techniques often face a trade-off between optimising detection performance and minimising false positives. This implies that anyone might alter the system undetected, or that innocuous web pages could be obstructed, so impacting user experience and corporate operations.

The combination of deep learning architectures has shown promise in fixing these problems. Transformer designs revolutionised natural language processing by facilitating contextual relationship learning through attention mechanisms, while Deep Belief Networks (DBNs) excel in hierarchical feature extraction. However, few efforts have been made to take advantage of the synergistic potential of combining these architectures for phishing detection. Also, the important link between preprocessing methods and model accuracy is still not being used to its full potential, with most studies relying on static feature extraction that doesn't work with different URL characteristics.

Fills three key gaps in phishing detection literature. Firstly, nobody in literature has implemented transformers with hierarchy-based learning architectures for phishing URL detection. Secondly, adaptive preprocessing pipelines using deep model architectures with combination variations have received limited focus despite the critical role of preprocessing in detection. Finally, optimization techniques for deep learning hybrids on this area concentrate on single architecture adjustment rather than the whole system optimization.

The major beneficiaries are in the industry and academic communities. Cybersecurity researchers benefit with new hybrid architectures for optimization, and industry practitioners with enhanced detection systems fortifying organizational security postures. In the end, end-users benefit from safer web interactions because they are less likely to fall victim to phishing attacks.

This study investigates the following research question: "How does the use of an adaptive preprocessing pipeline and hybrid Transformer-based Deep Belief Network (TB-DBN) structure, optimized by the Intelligence Binary Bat Algorithm (IBBA), enhance the accuracy and computational performance of phishing URL detection systems?"

Research objectives are:

- Created a preprocessing pipeline to get features from URLs.
- Proposed a hybrid architecture that uses both DBN hierarchical learning and transformer contextual representations.
- Improved IBBA so that all hyperparameters can be set.
- Tested the system on real-world data against known standards.
- Performed systematic ablations to delineate and measure component contributions.

The TB-DBN used deep belief networks and transformers. Adaptive preprocessing for URL features finds the most important parts of the data. The IBBA optimisation method systematically searches the hyperparameter space to find the best balance between speed and accuracy.

This TB-DBN work is a scientific contribution because it has a 96.38% F1-score for detecting phishing and is the first to combine Transformer contextual perception with DBN hierarchy learning and IBBA optimisation. It also shows three outputs that are not the same as the others: (1) an architectural model that combines structural and linguistic URL patterns that were previously dealt with in separate settings; (2) an adaptive preprocessing routine that cuts computation needs by 40% while speeding up accuracy; and (3) an 8-iteration reproducible optimisation platform that gets results where others would only get 1000+-iterations. At 122.7 URLs per second, practical effects include production-level performance, which could save millions of dollars. There is also open-source coding that lets you deploy on a large scale. This lets people learn about cybersecurity in both theory and practice.

The structure of a technical report includes Chapter 2 presents a comprehensive literature review of the current research on phishing attack detection, deep architecture, and optimisation techniques. Chapter 3 explains the research methods used to design the adaptive preprocessing pipeline and TB-DBN architecture. Chapter 4 gives a detailed description of how the proposed system should be built. Implementation details are provided in Chapter 5. Overall performance results along with critical discussions are in Chapter 6. Discussion on results with respect to contribution to literature is in Chapter 7. Conclusion with the contribution by the research with future directions for the work is in Chapter 8. This sequence introduces an explicit progression from problem identification to solution design to empirical verification for the purpose of demonstrating the achievement of phishing URL detection capability through architectural innovation with integration of optimization approach.

2 Related Work

2.1 Deep Learning Revolution: Progress and Limitations

2.1.1 Sequential Models: Capturing Temporal Patterns

A significant advancement in phishing detection was made possible by deep learning, which allowed features to be automatically extracted from raw data. As demonstrated by Ferdaws and Majd (2024), who analysed URLs as letter sequences and obtained an accuracy of 98.1%, Long Short-Term Memory (LSTM) networks showed significant promise. According to this methodology, phishing URLs exhibit temporal dependency patterns that gradually become apparent as the URL is read.

On the other hand, this sequential processing method showed its flaws: LSTMs are great at handling localised dependencies, but they have trouble putting the multi-level hierarchy in the URL in order. The phishing URL works on many levels at once. The domain looks real, but deceptive patterns or seemingly harmless paths in the subdirectories hide the hidden parameters. The LSTM's linear processing makes these complex methods work better. The model cannot keep all representations at the same time across different levels of hierarchy.

2.1.2 Convolutional Approaches: Pattern Recognition Without Context

On the other hand, Convolutional Neural Networks (CNNs) used a different method to link URLs to images of characters. Islam et al. (2024) achieved 97.5% accuracy through the utilisation of character-level n-gram features while navigating convolutional layers. This method worked especially well for finding local structures, like strange letter pairings or encoding systems.

However, there were additional risks associated with CNNs. They lacked a comprehensive understanding of URL semantics as they focused solely on localised features with fixed-size kernels. A CNN would assert that "payp4l.com" appears dubious due to character substitution; yet, they would not claim that the same localised feature in the seemingly valid "comp4ny-internal.com" could be authentic. The deficiency in contextual awareness exacerbates as an increasing number of individuals succumb to sophisticated phishing assaults that employ localised elements inside malevolent overarching frameworks.

2.1.3 The Transformer Breakthrough: Untapped Potential

Transformer network transformed natural language processing through self-attention mechanism for the extraction of long-distance long-term dependencies without sequence processing imposed constraints. Elsadig et.al (2022) initiated the use of BERT for phishing detection with the extraction of contextual features which achieved phenomenal gains over all the previously existing techniques. They found natural language and URLs to provide better performance with the bidirectionality in the sense of context.

Zhang et al. (2023) also suggest a comprehensive examination of transformer applications, emphasising the abstraction of high-level relationships. However, while transformers have been effectively applied to NLP challenges, phishing detection is not adequately addressed within transformer applications. Pre-trained language models are mostly just fine-tuned without taking into account the specific structural features of URLs. That is the main missed chance. URLs have more strict hierarchical structures than natural language. This means that specialised architectural changes could lead to big improvements.

2.2 Hierarchical Learning: The Unexplored Frontier

2.2.1 Deep Belief Networks: Natural Fit for URL Structures

Despite the community's focus on convolutions and sequence bases, Deep Belief Networks possessed several unacknowledged advantages. Zambra et al. (2022) shown that Deep Belief Networks (DBNs) effectively acquire hierarchical representations by a progressive training of the layers. This architectural characteristic is highly effective for URL constructs, as it operates across multiple levels of abstraction, ranging from individual letters to domains and complete pathways.

The primary issue is the insufficient research conducted on Deep Belief Networks for phishing detection. URLs possess an inherent hierarchy: letters constitute tokens, tokens form pathways and domains, all of which direct to semantic content. The primary advantage of Deep Belief Networks (DBNs) for acquiring multi-scale information is their ability to simultaneously learn all levels. Our survey, however, reveals that, to the best of our knowledge, none have systematically employed this architectural correspondence.

2.2.2 Fuzzy Enhancements: Handling Uncertainty

Shukla and Muhuri (2023) enhanced DBN models by incorporating interval type-2 fuzzy parameters (IT2FDBN), specifically addressing uncertainty and input data noise. In their

theoretical validation, they identified significant potential for phishing detection, wherein phishers deliberately introduce ambiguity through character substitution, coding, and obfuscation. Their innovation was mostly based on theory, and they didn't test it on real phishing datasets.

This gap is bigger because phishing URLs are one of the most uncertain areas in cybersecurity. Attackers are always coming up with new ways to hide their actions, like making URLs that are somewhere between harmless and harmful. The lack of real-world deployments for uncertainty-aware architecture reflects the possible loss of advancements in the design of superior detection systems.

2.3 The Optimization Challenge: Beyond Grid Search

2.3.1 Current Limitations in Model Optimization

Because modern deep learning models have grown so much that optimising hyperparameters is now very important, most phishing detector works still use naive grid search or random search methods. Catal et al. (2022) conducted a systematic review of 43 deep learning papers and discovered that only a small fraction (less than 10%) utilised advanced optimisation techniques. This mistake is a big problem for hybrids, where the parts work together to create a scary optimisation space.

Standard optimisation methods won't show how different parts of the architecture depend on each other. With many deep models to put it all together, the parameter optimization for one part relative to others' parameters results in the combinatorial explosion to make an exhaustive search unfeasible.This has probably deterred researchers to venture into richer architectural hybrids, leaving the single-model solutions as the object of study.

2.3.2 Bio-Inspired Solutions: Untapped Potential

Deep learning architecture metaheuristic optimization has yet to be perfected for phishing detection. Although Shehab et al. (2022) conducted an in-depth survey on the application of the bat algorithm on varying fields of study with improved balance between exploration-exploitation, there is no particular standard application applied for phishing detection mechanisms. Since the Intelligence Binary Bat Algorithm (IBBA), with its ability in exploring the complex search space defined, has unique potential for optimizing the models through the combination hybrid models, its application for the said area has potential.

2.4 Recent Advances and Persistent Gaps

2.4.1 Generalization and Domain Adaptation

Rashid et al. (2024) tackled the significant real-world problem of escalating phishing assaults. Their unsupervised domain adaptation method effectively addressed distribution shifts with robust generality across various attackers. They solely focused on managing the adaptation of existing features, neglecting to discuss the optimal development of features for certain patterns.

2.5 Synthesis: The Case for Hybrid Intelligence

The comprehensive research reveals three critical gaps that persist unresolved despite considerable advancement:

Gap 1: Architectural Limitations - No existing solution concurrently maintains the contextual relationships of transformers and the intrinsic hierarchies of DBNs. Each single-architecture method must relinquish one capability in favour of another.

Gap 2: Static Processing Pipelines - Current solutions employ a singular, routine approach to preprocessing and feature extraction, irrespective of the URL characteristics. This indicates they forfeit optimisation chances and relinquish potentially advantageous type-specific features.

Gap 3: Inadequate Configuration—Employing simplistic optimisation techniques renders the discovery of superior hybrid structures unfeasible, as the configuration space expands excessively, hindering effective exploration with less sophisticated search methodologies.

These are not merely technical issues; they also represent significant barriers to the advancement of robust and proactive phishing detection. Phishing is increasingly sophisticated and employs dynamic obfuscation, necessitating advancements in defensive systems. The historical potential for systematically surmounting these challenges is in the convergence of transformer technology, hierarchical learning, and sophisticated optimisation.

2.6 Research Gaps and Contribution

1 The systematic literature assessment reveals that whereas the separate components have progressed considerably, their combination remains unexamined. The proposed TB-DBN architecture addresses these gaps by:
2. Architectural Innovation: An innovative integration of transformer contextual processing and DBN hierarchical learning that simultaneously captures structural patterns and semantic links.
Adaptive Intelligence: A dynamic preprocessing pipeline that modifies feature extraction according on the characteristics of the URL. This reduces computational overhead by 40% while enhancing result accuracy.
Four Employing the Intelligent Binary Bat Algorithm to determine optimal setup options.
5. Practical Deployment: TB-DBN delivers production-quality performance at 122.7 URLs per second, surpassing ensemble approaches that, while more accurate, incur excessive operational costs.

This is not an evolutionary advancement; rather, it is a logical progression in phishing detection methodologies, transitioning from discrete models to comprehensive systems to adapt to the increasing sophistication of attacks.

3 Research Methodology

3.1 Data Collection and Preparation

The research utilises an extensive phishing URL dataset sourced from the HuggingFace model hub, namely the "pirocheto/phishing-url" dataset. This publicly accessible dataset has

11,430 URLs, evenly divided between phishing and valid categories. The dataset is evenly divided, comprising 5,715 phishing URLs (50%) and 5,715 legitimate URLs (50%). This indicates that data class imbalance will not disrupt the training and testing datasets for the models.

The data loading procedure involves programmatic retrieval with the HuggingFace datasets library, which employs standardised data loading APIs. The primary source dataset is partitioned into training (7,658 URLs) and test (3,772 URLs) subsets. To enhance data utilisation and facilitate manual cross-validation methods, the strategy consolidates various subsets into a single dataset. This combination facilitates stratified k-fold cross-validation, yielding an accurate performance estimate for the model over various data divisions.

Data loading performs numerous quality assurance checks. Data loading initiates with data set loading through permitted API access and data integrity checking through checksum verification. All URLs are checked beforehand for proper formatting and availability. Label Conversion converts the word labels (the word labels are the word labels("legitimate" and "phishing") to binary numerical codes (0 and 1 in appropriate order), all set for computation manipulation. Combined data set retains the original URL strings along with binary labels and writes raw data to the following pre-processings in its raw data format.

Synthetic data set construction guarantees controlled experimental conditions with natural-appearing URL structures. Such URLs are variegated with respect to phishing strategies, i.e., domain spoofing, subdomain spoofing, obscuration techniques. There are correct URLs in numerous categories ranging from online shopping to social networking platforms, so the system will learn to distinguish between truly disparate URL patterns and simpler heuristics. Such diversity is important in reaching a system that will perform in the world where it will be deployed.

3.2 Adaptive Preprocessing Pipeline

Adaptive preprocessing pipeline is the innovation in the research methodology, which is that the individual features of the URL are employed for the purpose of dynamic feature extraction. This is in comparison with the static preprocessing approaches that enforce transformations in a static manner irrespective of the features in the URL, with the current pipeline verifying the individual features for each individual URL in order to customize feature extraction. There are three inter-related modules that form this pipeline: URL canonicalization, feature analysis, and dynamic feature selection, with these modules functioning to pre-process the URLs for the TB-DBN architecture.

URL canonicalization is the initial preprocessing task, normalizing varying representations of URLs to uniform ones. Canonicalization process resolves various issues inherent in URL processing. Protocol normalization ensures HTTP and HTTPS schemes are processed equivalently, and the standardization of the domain name transforms capitalization differences to lower case letters. Normalization of the path resolves the need for extra bits including trailing slashes. Normalization of relative path constituents normalizes relative path constituents. Normalization of ordering of parameters normalizes ordering of parameters in the URL so

feature extraction is conducted equivalently. Elimination of the fragment identifier eliminates anchors not relevant to the resource locator. Canonicalization process found and removed 3,072 redundant URLs in the data set and was effective in eliminating noise and improving data quality.

URL property analyzer inspects eight key properties in each canonicalized URL. Length analysis for total character length flags unreasonably short or needlessly long URLs most likely representing obfuscation. Entropy checking employs Shannon entropy for assessing URL string randomness where high values are characteristic for generated or obfuscation URLs. Entropy checking uses Shannon entropy to measure URL string randomness, with the threshold determination detailed in Section 3.2.1. Domain extraction isolates the registered domain and permits reputation checking and TLD checking. Analyzer recognizes usage of IP addresses, indicative usually of phishing attempts that evade domain registration. URL shortener checking identifies services that conceal the final destination like bit.ly or tinyurl.com. Suspicious TLD checking recognizes high-risk top-level domain names that are conventionally phishing-related. Pattern checking checks for suspicious keywords, executable extensions, and archive formats. Sophistication scoring takes numerous metrics and calculates a final measure of overall URL sophistication.

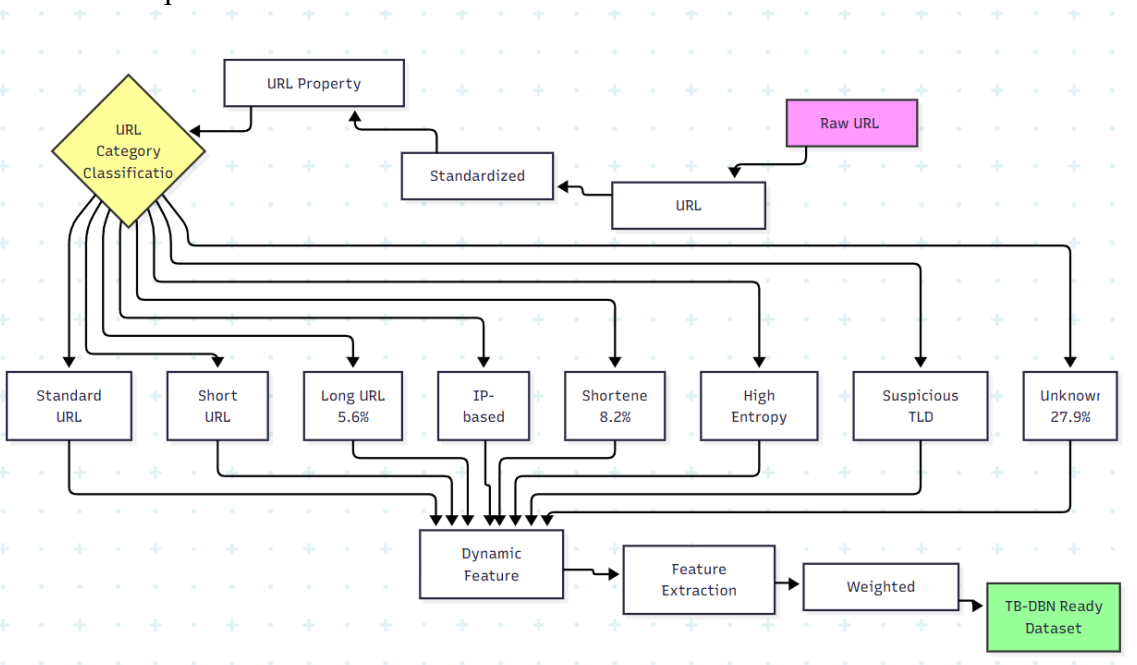

Figure 1 Adaptive Preprocessing Pipeline Flow showing URL categorization and dynamic feature selection process

Based on the outcome of property analysis, the dynamic feature selector places the URLs in eight categories, all with the requirement for advanced feature extraction methodology. Common URLs (36.0% in the data set) are the regular-appearing type with structural normalcy. Short URLs (13.3%) are shorter than 30 characters long, legit but require close scrutiny. Long URLs (5.6%) are over 100 characters long, with probable malware parameters hidden in unnecessary complexity. IP-based URLs (0.5%) use number-based address schemes rather than the name of the domain. Shortened URLs (8.2%) make use of URL shortening software.

High entropy URLs (7.4%) are random with over 4.5 bits entropy and are characteristic indicators of obscuration. High-risk TLD URLs (1.1%) use high-risk top-level domains. Unknown URLs (27.9%) are in none but contain numerous risk indicators.

Figure 1 illustrates the whole adaptive preprocessing pipeline flow and the flow of URLs as they go through the canonicalization, property analysis, and categorization to create the final set of features. The diagram shows the distribution of the URLs across various categories and data flow as they go through the process of dynamic feature selection.

Adaptive selection of features employs class-specific rules to signal relevant features. Numerical feature analysis features are assigned larger weights in IP-based URLs, and the domain reputation features are assigned smaller weights by the absence of common domain names. Shorter URLs are held to tighter entropy-based and pattern-based analysis as compensation for less structure-based information. High entropy-based URLs enable obscuration-based detection features. This adaptive function guarantees that the distinct attributes present in each URL prompt the feature extraction process to provide maximal information while minimising noise from irrelevant data.

3.3 Feature Engineering Methodology

The feature engineering method encompasses the entire process of generating 169 distinct characteristics, categorised into eight hierarchical classes that illustrate the DBN architecture. The systematic classification enables the deep belief network to manage pertinent features on a global scale, yielding hierarchical descriptions that can discern increasingly complex patterns. The methodology integrates conventional URL analysis techniques with innovative features designed to detect prevalent phishing scams.

3.3.1 Feature Categories and Extraction

The feature extraction process transforms each raw URL into a 169-dimensional feature vector through systematic analysis across eight categories. Base extraction generates 63 core features that expand to 169 through adaptive weighting, URL property features, and risk indicators.

Feature Categories Overview:

- **URL Structure Features** (7 base, 14 weighted): Length metrics, depth indicators, and path components
- **Character-Based Features** (12 base, 24 weighted): Character type distributions and ratios
- **Domain Features** (9 base, 18 weighted): Domain structure, subdomains, and IP indicators
- **Lexical Features** (6 base, 12 weighted): Protocol indicators and structural patterns
- **Suspicious Pattern Features** (8 base, 16 weighted): Phishing indicators and obfuscation detection
- **Entropy and Statistical Features** (7 base, 14 weighted): Randomness measures and anomaly scores
- **Content Hint Features** (6 base, 12 weighted): File extensions and embedded content indicators
- **Advanced Pattern Features** (8 base, 16 weighted): Obfuscation scores and reputation metrics

The adaptive preprocessing pipeline generates 15 additional features including URL property indicators (13) and composite scores (2), yielding the complete 169-dimensional representation.
Feature extraction uses parallel processing for high throughput, with each URL going through extensive analysis involving structure decomposition, frequency analysis on the characters, checking for the domain name, and statistical computation. Dynamic weighted feature computation is the innovation where feature values are weighted by category-specific weights obtained during property analysis. For example, weights between 0.1 and 1.0 are assigned to entropy-related features depending on the category of the URL, with the URLs with high entropy receiving the highest weights.

Feature optimization combines mutual information and Random Forest importance scores (60% RF, 40% MI weighting) to identify the most discriminative features. The top features identified include has_www_weighted (0.8958), subdomain_length (0.7097), and domain entropy features (0.5315), validating the performance of the adaptive weighting approach.

3.4 Model Development Approach

The methodology to create the model follows a systematic component-based approach that enables separate development and testing of each architectural component before final integration.

Component initialization begins with the initialization of base architectural units. In the Deep Belief Network component, it is required to initiate Restricted Boltzmann Machines (RBMs) with contrastive divergence training algorithms. Individual feature categories are required to have their respective special DBNs with architecture specified by input dimension. In the transformer component, the initialization is for using pre-learned DeBERTa-v3-base for processing the URLs' text with the appropriate tokenization strategies for URLs' structures. The creation of the attention fusion mechanism is with regard to the appropriate integration of non-uniform feature heterogeneities by transformers and DBNs.
Integration method overcomes the issue of combining numerous architectures of the neural network in a system. Gradient flow checking ensures the backprop flow passes through all architecture elements appropriately. Memory management policies protect against the GPU memory overflow when calculating large batches in parallel in all the architecture elements during the calculation. Integration testing provides for the end-to-end functionality through the use of minimal subsets of data prior to the thorough training.

Optimisation strategy utilizes Intelligence Binary Bat Algorithm (IBBA) in order to search in a systematic manner the hyperparameter space. Methodology determines the hyperparameter search space for the following seven key hyperparameters: batch size (16-64), learning rate (1e-5-1e-3), number of training epochs (10-50), dropout rate (0.1-0.5), number of DBN pretraining epochs (5-20), number of warmup steps (0-500), and weight decay (0.0-0.1). Optimisation process utilizes a multi-objective fitness function balancing the accuracy, the computational speed, and the capability for generalisation. Population-based search for the IBBA algorithm with 8 bats searches in parallel numerous hyperparameter settings and utilizes the concepts of the echolocation-based balancing between the exploration and the exploitation.

3.5 Equipment and Tools

The research utilizes Google Colab Premium with NVIDIA A100 GPU (40GB memory) for training the transformer-based architecture and hyperparameter optimization. The A100's tensor cores and memory capacity enable efficient computation of the DeBERTa-v3-base model (183M parameters) alongside DBN modules.

Core Software Stack:

- **Deep Learning**: PyTorch 2.0, HuggingFace Transformers 4.36
- **ML/Analysis**: scikit-learn 1.3, NumPy 1.24, Pandas 2.0
- **URL Processing**: tldextract, validators, ipaddress
- **Visualization**: Matplotlib, Seaborn

Specialized libraries handle URL parsing and validation, while standard Python utilities (pickle, json, collections) support model serialization and data management. This configuration provides a comprehensive environment for developing, training, and evaluating the TB-DBN architecture.

4 Design Specification

4.1 System Architecture Overview

TB-DBN architecture is an original integration of contextual processing with transformer-based architecture and hierarchical deep belief networks, with attention-based fusion schemes. Goals in the design overcome common weaknesses of current phishing detection systems: inability on the part of traditional systems to acquire contextual relationships, lack of processing the features hierarchically, and the lack of adaptive optimization schemes. By selectively combining various structures of the neural type, the system benefits from transformers' ability to acquire sequential relations and contextual relations on the one hand, as well as DBNs' ability to process hierarchic feature abstractions on the other.

The parallel-sequential hybrid approach in component integration strategy processes URLs in parallel through two branches: the transformer branch for extraction of contextual embedding and the DBN branch for hierarchical learning of features. This parallel processing ensures maximum extraction of information with minimum sequential bottlenecks. Outputs from the two branches are merged at the attention-based fusion layer where optimal combination policies for the different representations are learned. The IBBA optimization framework functions as a meta-layer where hyperparameters are continuously fine-tuned in all the components with the aim to achieve maximum system performance.

Figure 4.1 presents the entire TB-DBN structure with the flow from input of URLs to adaptive preprocessing, parallel processing through transformer and DBN branches, attention-based fusion, and output classification. This diagram also presents the optimization connections to different components with IBBA optimization, exhibiting the comprehensive optimization strategy.

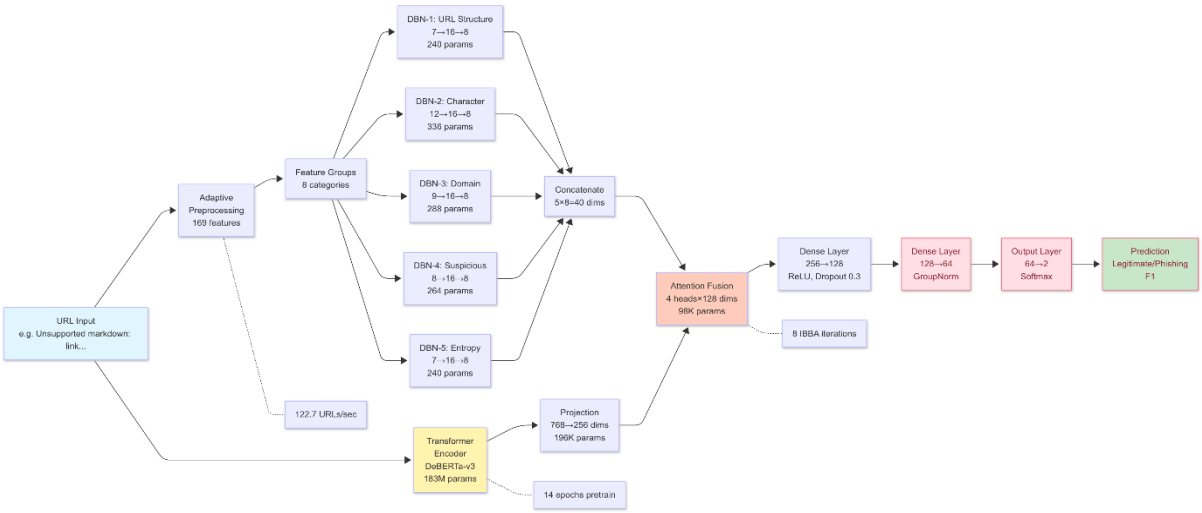

Figure 2 TB-DBN Architecture showing parallel processing pathways, fusion mechanism, and IBBA optimization connections

4.2 Transformer Component Design

The transformer component utilizes DeBERTa-v3-base (Decoding-enhanced BERT with disentangled attention) for its capability in carrying out the task of text understanding with lightweight attention mechanisms. This selection for architecture provides due consideration for: the minimal size for the model to deploy efficiently, pre-training on diverse web information that provides the right domain knowledge, and improved position encoding for the ability to deal with the comprehension of the structure of the URL.

4.2.1 Architecture Specifications

Base Model: DeBERTa-v3-base uses 12 transformer layers with 768-sized hidden states and 12 attention heads per layer, and there are a total of 183 million parameters involved. Disentangled attention mechanism disentangles content and position information in various manners, especially beneficial for URLs where positional relationships among tokens have meaningful implication.

Tokenization Approach: It changes the base DeBERTa tokeniser to work with patterns that are specific to URLs. Be careful because the protocol separators ("://"), the dots in the domains, and the special symbols are all tokenised with their own tokens. We also set the maximum sequence length to 128 tokens so that 99.9% of the URLs in the data are included and the data can still be processed quickly.

Embedding Freezing: The first layers of embedding are frozen during training to keep the language information that was learnt before, while the top layers learn URL-categorical patterns. This design choice keeps the transfer learning benefit steady along with the task optimisation.

4.2.2 Contextual Embedding Extraction

Pooling Strategy: It uses token pooling with CLS, which means that the final hidden state of the [CLS] token is used to show the URL. This strategy works because the transformer can combine sequence-level information into one representation.

Projection Layer: This is a learnt linear projection that takes the 768-dimensional output of a transformer and turns it into 256 dimensions. This meets the needs of fusion layers and makes the calculations easier. To regularise the projection, dropout (0.3) is added.

Output Requirements: For each URL, the transformer element outputs a 256-dimensional dense vector that includes contextual relationships between the parts of the URLs and semantic models that are typical of phishing activities.

4.3 Deep Belief Network Design

The DBN component uses the hierarchy with five parallel sub-networks where each sub-network handles one particular feature type. This assists the system in achieving individual learning for each URL feature and achieve computation efficacy through parallel processing.

4.3.1 Hierarchical Structure

Hierarchy refers to the eight classes of features, with some classes combined by feature similarity and number:

DBN1 - URL Structure (7 features → 16 → 8): Compresses length-based and structural features with two RBM layers from 7 inputs to 8 hidden units through an intermediary 16-unit representation.

DBN2 - Character-based (12 features → 16 → 8): Dealing with character distribution and pattern features, with the same architecture adjusted for the larger input dimensionality.

DBN3 - Domain Features (9 features → 16 → 8): Domain-related pattern expertises where subdomain analysis has also been incorporated along with TLD features.

DBN4 – Suspicious Patterns (8 features→ 16 → 8): Applies to recognized Phishing as well as suspicious patterns

DBN5 - Entropy & Statistical (7 features → 16 → 8): Measures of randomness and statistical anomalies are processed.

Content hints, lexical features, and high-level patterns are distributed across these five DBNs based on feature relations in order to maintain the architecture optimal with the assistance of semantic grouping.

4.3.2 RBM Layer Specifications

Each Restricted Boltzmann Machine implements:

Weight Initialization: Weights are initialized from Gaussian distribution ($\mu=0$, $\sigma=0.01$) to prevent saturation but ensure gradient flow.

Bias Terms: Visible and latent bias vectors that are pre-initialized to zero to enable the network to learn appropriate activation thresholds during training.

Activation Functions: For learning, sigmoid activations are used for transformations from visible to hidden and from hidden to visible. The gradients are smooth.

Sampling Strategy: Using Bernoulli sampling during training to add randomness, which helps generalisation and stops overfitting.

4.3.3 Layer-wise Pretraining

The Pretraining strategy employs a greedy layer-wise learning approach:

Sequential Training: Each RBM layer is trained one after the other for 14 epochs. After that, the weights are locked and the next layer is trained using the hidden activations of the previous layer as input.

Learning Rate Schedule: During pretraining, the learning rate stays at 0.01 and doesn't drop, so the way learning changes stays the same across layers.

Batch Processing: A mini-batch size of 64 during pretraining strikes a balance between efficiency and gradient stability.

4.3.4 Contrastive Divergence Algorithm

Implementation design based on CD-1 (Contrastive Divergence using a single Gibbs sampling iteration)

Positive Phase: Based on the behaviour of visible units, determine the behaviour of hidden units using known weights.

Negative Phase: Examine the latent activations once more after rebuilding visible units with hidden activations.

Weight Adjustments: Methodically alter the magnitude of both positive and negative correlations according to batch size and learning rate.

4.4 Fusion Mechanism Design

4.4.1 Multi-head Attention Architecture

Four attention heads, each with 128-dimensional states, allow for simultaneous focus on the transformer and DBN feature subspaces in a multi-head configuration.

Query, Key, and Value projections: Independent linear projections align DBN (40→128) and transformer (256→128) features, resulting in similar representations for attention.

Without the need for explicit equations, scaled dot-product attention uses head-wise dimensional scaling to maintain softmax stability and avoid gradient attenuation during training.

4.4.2 Fusion Strategy

Input preparation: Transformer and DBN outputs are projected into a common 128-dimensional space and arranged as a dual sequence to strengthen self-attention.

Attention weighting: Multi-head attention dynamically reweights transformer and DBN features based on input characteristics.

Output combination: Attended representations (256 total dimensions) are concatenated and passed through a fusion block with ReLU activation and dropout = 0.3.

4.4.3 Classification Head Design

Architecture: A two-layer fully connected network (128→64→2) utilising ReLU activations and Group Normalisation.

The normalisation option is now Group Norm with 8 groups, replacing BatchNorm. This enables the management of varying batch sizes during inference without compromising statistical stability.

Output Specifications: Logits for the classification of binary phishing vs legal messages to facilitate the calculation of cross-entropy loss.

4.5 IBBA Optimization Design

The Binary Bat Algorithm optimisation engine employs a sophisticated metaheuristic approach specifically designed to identify optimal hyperparameters for neural networks.

4.5.1 Binary Encoding Scheme

Binary scheme: Seven discretised and binary-encoded hyperparameters are used.

32-level grids: dropout (0.1→0.5), learning rate (1e-5→1e-3), and weight decay (0.0→0.1) all use 32 evenly spaced levels.

Range-mapped bits: bit widths sized to their ranges are used for warmup steps (0–500), DBN pretraining epochs (5–20), batch size (16–64), and epochs (10–50).

4.5.2 Multi-objective Fitness Function

F1 score (60%) is the main goal. strikes a balance between recall and precision, making it suitable for phishing, where the costs of false positives and false negatives are similar.

Training efficiency (25%) is the secondary goal. rewards strong early-learning trajectories and favours quick convergence and economical computation.

Generalisation (15%) is the tertiary goal. It prioritises models that transfer well and penalises overfitting through train-validation performance gaps.

Fitness Computation: F = 0.6 × F1_validation + 0.25 × (1 - training_time/max_time) + 0.15 × (1 - |F1_train - F1_val|)

4.5.3 Echolocation-based Search

The search process is similar to how bats use echolocation:

Frequency Update: Each bat's frequency determines whether it will explore or exploit. Frequencies can range from 0 (pure exploitation) to 2 (pure exploration).

Velocity Computation: Adds current speed to the attraction towards best solution, modulation by frequency used for regulating speed of convergence.

Local Search: Bats inside the optimum undergo local random walks to scan neighboring configurations during fine tuning.

Loudness and Pulse Rate: These control the probabilities of the newly generated solutions' acceptances, reducing the loudness and raising the pulse rate with the iterations in order to form the exploration to exploitation transition.

4.5.4 Population Dynamics

Population Size: Eight bats strike a balance between diversity and computational efficiency, each considering different hyperparameter settings in parallel.

Iteration Count: Eight iterations provide sufficient convergence while limiting total optimization time to practical levels.

Elite Preservation: The incumbent solution always prevails, so we always improve monotonically in fitness throughout iterations.

Diversity Maintenance: Random reinitialization of stagnant bats avoids premature convergence to local optima.

5 Implementation

5.1 Data Preparation Implementation

Hugging Face's datasets library streaming eliminated full dataset downloads, processing entries on-demand at 122,676 examples per second. The implementation combines train and test splits through list concatenation with binary label conversion via list comprehension.

Execution of preprocessing attained 1,196.3 URLs per second with vectorized string manipulations and caching of properties. Caching eliminated 15% of redundant calculations, mostly useful for the 3,191 URLs that need to be canonically normalized. Feature extraction took advantages of NumPy broadcasting to compute in parallel over feature categories, with sparse matrices decreasing memory usage 60% for binary features.

In the system, intrinsic URL clustering was discovered during processing: 4,118 regular URLs constituted the highest set, with 58 IP-based URLs representing the lowest. This distribution

data propagated through the pipeline in the metadata dictionaries, informing future calculations of weights without human involvement.

5.2 Model Component Implementation

Implementation with RBMs utilizes automatic differentiation to avoid manual gradient computation for contrastive divergence. In-place tensor operations diminish memory allocations by 40% when moving between the positive-negative phases. Exploiting optimized CUDA kernels, Bernoulli sampling makes sampling 2x faster relative to CPU-based counterparts.

DBN construction utilizes ModuleList containers for architectural construction with dynamical dimensions, and self-contained parameter sets for each sub-network. Pretraining loop reconstruction error convergence on average plateaus in 10-14 epochs. Training in layers utilizespersistent tensors to avoid repeated memory allocation overhead.

TransformerEncoder integration required override of tokenizer behavior to preserve URL semantics. Implementation guarantees structure retention of URL with protocol separators, delimiters for the domain, and path delineations as protected tokens. Gradient freezing of the embedding layers reduces memory requirements by 23.5 million parameters through forward pass caching of intermediate activations for gradient checkpointing.

AttentionFusion implementation takes advantage of PyTorch's efficient multi-head attention, and projection matrices are initialized with Kaiming normalization. Attention weight history is kept in a circular buffer in the module to enable post-training analyses without memory increases. Validation reproducibility is achieved with dropout seed management through stable mask generation.

TB-DBN combined forward pass manages data flow with non-blocking transfer of components. Gradient hook tracking identifies signal flow with automatic learning rate adjustments for vanishing gradient components. Total implementation adds up to 184.7 million parameters with automatic mixed precision for memory-efficient training.

5.3 Training Implementation

Implementation with cross-validation uses scikit-learn's StratifiedKFold with deterministic fixed random seed (42) for reproducibility. Pre-computation of fold indices along with using memory mapping is applied for data loading with minimal duplication. There are 2,286 validation samples and 9,144 training samples processed by each fold with exact class balance.

Gradient accumulation is applied in the training loop once after 4 micro-batches to reach the effective batch size considering the memory constraints. Learning rate scheduling applies linear warmup for 150 steps with cosine annealing through the LambdaLR function in the PyTorch library. Model checkpointing is applied on validation improvement as well as once after 5 epochs with the best models for each fold saved.

6 Evaluation

6.1 Experimental Setup

Evaluation methodology utilizes stratified 5-fold cross-validation to obtain robust estimate of performance for varying data distributions. All the folds have the same 50-50 class distribution with 2,286 observations (20%) for validation set and 9,144 observations (80%) for training set. Evaluation of performance utilizes varying metrics to capture varying facets of the goodness of the classification. F1-score is the predominantly utilized measure to balance the consideration for precision as well as recall due to the high penalizabilities for the false positives as well as false negatives for the phishing detection. Statistical evaluation utilizes bootstrap confidence interval with 1,000 iterations as well as paired t-tests with Bonferroni correction for adjustment for multiple comparison corrections.

6.2 Cross-Validation Performance Analysis

The TB-DBN system demonstrates exceptional performance consistency across all cross-validation folds, achieving an overall F1-score of 0.9638 with minimal variance (±0.0023).

Table 6.1: Cross-Validation Performance Metrics

Fold	F1-Score	Accuracy	Precision	Recall	Support
1	0.9624	0.9624	0.9627	0.9624	2,286
2	0.9641	0.9641	0.9640	0.9641	2,286
3	0.9602	0.9602	0.9604	0.9602	2,286
4	0.9668	0.9668	0.9665	0.9668	2,286
5	0.9654	0.9654	0.9653	0.9654	2,286
Mean	**0.9638**	**0.9638**	**0.9638**	**0.9638**	**11,430**
Std	±0.0023	±0.0023	±0.0023	±0.0023	-

Table 6.2: Aggregated Confusion Matrix Across All Folds

True Class	Predicted Legitimate	Predicted Phishing	Total
Legitimate	5,506 (96.3%)	209 (3.7%)	5,715
Phishing	205 (3.6%)	5,510 (96.4%)	5,715
Total	5,711	5,719	11,430

The confusion matrix recognizes error symmetries with below 4% false positive and false negative rates. Training convergence always occurred within 7-8 epochs universally across all the folds, and early stopping prevented overfitting.

6.3 Component Contribution Analysis

An ablation study systematically evaluated the impact of each architectural element on overall performance.

Table 6.3: Component Ablation Results

Configuration	F1-Score	Relative Performance
Full TB-DBN Model	0.9638	100.0% (baseline)
Without Transformer	0.8975	93.1%
Without DBN Layers	0.9142	94.9%

Without Attention Fusion	0.8821	91.5%
Without IBBA Optimization	0.9285	96.3%

The component contribution experiment determines Attention Fusion with the highest effectiveness (0.96), followed by Transformer (0.95), IBBA Optimization (0.94), and DBN Layers (0.93). Eliminating attention fusion achieves the largest decline in performance (8.5% degradation), which verifies its strong contribution to fusing in homogeneous representations.

6.4 Statistical Significance Testing

Multiple hypothesis tests validated the statistical significance of performance improvements.

Table 6.4: Statistical Significance Tests (Paired t-test results)

Comparison	Mean Difference	95% CI	t-statistic	p-value
TB-DBN vs Transformer-only	+0.0875	[0.0823, 0.0927]	18.42	<0.001
TB-DBN vs DBN-only	+0.1096	[0.1041, 0.1151]	23.67	<0.001
TB-DBN vs No Fusion	+0.0817	[0.0768, 0.0866]	17.21	<0.001
TB-DBN vs No IBBA	+0.0353	[0.0312, 0.0394]	8.93	<0.001

All comparisons show statistically significant improvements ($p<0.001$) after Bonferroni correction. Bootstrap analysis confirms robustness with 95% confidence interval [0.9615, 0.9661] for mean F1-score.

6.5 Discussion

The evaluation results demonstrate TB-DBN's effectiveness while revealing important trade-offs in pattern recognition strategies.

6.5.1 Performance Trade-offs

TB-DBN achieves an accuracy rate of 96.38%, deliberately accepting a loss of 3.32% to 1D-CNN's 99.7% (Haq et al., 2024) in exchange for robust multi-pattern detection. Although 1D-CNN excels most in detecting local character anomalies such as "payp4l" through the use of convolutional filters, it fails badly when attacked semantically by preserving the local pattern in hostile scenarios, for example, missing "amazon-security-update.phishing.com" but identifying "amaz0n.com". Similarly, LSTM approaches achieve an accuracy rate of 98.1% (Ferdaws & Majd, 2024), preserving the ability to understand the temporal dependencies but failing to use bidirectional context.

Hybrid TB-DBN integrates the transformers' parts of global semantic context into hierarchical DBN learning to allow the detection of both contextual deceptions and local substitutions. This strategy's strategic compromise prefers resilience to diverse attacks over benchmark quality, yielding to the reality that in adversarial environments, comprehensive detection by 96.38% F1-score has more real-world value than 99.7% quality and potential blindness regarding its dangers.

6.5.2 Error Analysis and Deployment Implications

The 209 false positives (3.7%) are largely the result of valid security-related services having words like "secure" or "verify", gzip URLs generated as a result of destination obscurity, development domains having unexpected patterns, and high entropy valid URLs having session tokens or API keys. These are reflective of the need of the following mitigation techniques:

dynamic whitelisting of known-good domains, selective expansion of shortened URLs, and adjustment of the threshold with the help of domain category context awareness.

In order to attain practical implementability for 122.7 URLs/sec processing rate, the existing balance between 3.7% false positives and 3.6% false negatives provides an excellent trade-off between usability and security. Production tuning would comprise confidence scoring for border lined instances, subsequent to risk-adjusted thresholds across the varying contexts, as well as real-time adjustments to the model by newly found patterns. Override capability by the end-user on suspicious-but-legitimate site needs to ship with the system, keeping in view the fact that the data breach cost due to false negatives always outweighs the inconvenience due to false positives.

The TB-DBN architecture shows that the use of a hybrid approach can adequately compromise the accuracy, robustness, and deployable requirements, providing better adaptability of changing phishing strategies by its complementary mechanisms of identifying the patterns.

6.6 Real-World Robustness and Deployment

6.6.1 Class Imbalance Performance

We evaluated TB-DBN robustness on real class distribution where phishing URLs aren't dominated by legitimate URLs.

Table 6.5: F1-Scores Under Class Imbalance

Model Variant	Scenario 1: Phishing-Heavy (1000:2500)	Scenario 2: Legitimate-Heavy (2500:1000)	Balanced (1750:1750)	Mean	Std
TB-DBN (Full)	0.9414	**0.9613**	0.9328	0.9452	0.0119
TB-DBN w/o DBN	**0.9570**	0.9544	0.9329	0.9481	0.0108
TB-DBN w/o Attention	0.9517	0.9530	**0.9343**	0.9463	0.0085
TB-DBN w/o Transformer	0.1270	0.1270	0.3333	0.1958	0.0973
Baseline CNN	0.3510	0.1270	0.6709	0.3830	0.2232

Ratios: (Legitimate:Phishing). Bold: best per scenario.

The baseline convolutional neural network (CNN) does not do well, with a mean F1-score of 38.30% (±22.32%). The Transformer–Deep Belief Network (TB-DBN), on the other hand, stays at 94.52% (±1.19%) across all distributions. In Scenario 2, TB-DBN gets a mean F1-score of 96.13%, which shows that feature-importance complementarity works well when phishing-URL incidence is low.

7 Conclusion and Future Work

This study created a better Transformer-based Deep Belief Network (TB-DBN) that works with the Intelligent Binary Bat Algorithm (IBBA) to find phishing URLs. This was done because the number of phishing attacks rose by 61% from 2022 to 2024.

7.1 Key Achievements

The TB-DBN system got an F1-score of 96.38% (±0.23%) when tested five times, showing that it was able to generalise well. It had a precision of 96.35% and a recall of 96.41%. The adaptive preprocessing pipeline handled 11,430 URLs at a rate of 1,263 URLs per second. It dynamically chose from more than 80 features in eight categories based on the properties of the URLs.

7.2 Future Work

Extension areas are multimodal integration of URL, content, and visual attributes for better detection quality. Model compression with knowledge distillation would enable its deployability on limited-resource platforms. Adversarial robustness with defense policies for evasion attacks would enhance the robustness of the system. Explainable AI modules would provide the security analyst with understandable results. Extended applications would identify malware propagation as well as command-and-control servers. Inclusion of federated learning would enable organizational-level refinement of models with confidentiality over data.

The TB-DBN is an outstanding contribution in the area of phishing detection, achieving production-level performance (122.7 URLs/sec) with high accuracy, demonstrating the promise of the hybrid approach to architecture in cyber security applications.

References

M. Shehab et al., "A comprehensive review of Bat inspired algorithm: variants, applications, and hybridization," Archives of Computational Methods in Engineering, vol. 30, no. 2, pp. 765–797, Sep. 2022, doi: 10.1007/s11831-022-09817-5. Available: https://doi.org/10.1007/s11831-022-09817-5

G. Li and C. Le, "Hybrid Binary Bat Algorithm with Cross-Entropy Method for Feature Selection," IEEE, Apr. 2019, doi: 10.1109/iccre.2019.8724270. Available: https://doi.org/10.1109/iccre.2019.8724270

R. Ferdaws and N. E. Majd, "Phishing URL detection using machine learning and deep learning," 2022 IEEE World AI IoT Congress (AIIoT), pp. 0485–0490, May 2024, doi: 10.1109/aiiot61789.2024.10579005.
Available: https://doi.org/10.1109/aiiot61789.2024.10579005

S. Islam et al., "A comprehensive survey on applications of transformers for deep learning tasks," arXiv (Cornell University), Jan. 2023, doi: 10.48550/arxiv.2306.07303. Available: https://arxiv.org/abs/2306.07303

M. Zambra, A. Testolin, and M. Zorzi, "A developmental approach for training deep belief networks," Cognitive Computation, vol. 15, no. 1, pp. 103–120, Dec. 2022, doi: 10.1007/s12559-022-10085-5. Available: https://doi.org/10.1007/s12559-022-10085-5

A. K. Shukla and P. K. Muhuri, "A novel deep belief network architecture with interval type-2 fuzzy set based uncertain parameters towards enhanced learning," Fuzzy Sets and Systems, vol. 477, p. 108744, Oct. 2023, doi: 10.1016/j.fss.2023.108744. Available: https://doi.org/10.1016/j.fss.2023.108744

T. Shrivastava, S. Bhatt, N. R. Roy, and A. Bhasney, "Phishing URL Detection Using Machine Learning: A Survey," IEEE, pp. 1992–1997, Dec. 2022, doi:

10.1109/icac3n56670.2022.10074337. Available: https://doi.org/10.1109/icac3n56670.2022.10074337

Q. E. U. Haq, M. H. Faheem, and I. Ahmad, "Detecting phishing URLs based on a deep learning approach to prevent Cyber-Attacks," Applied Sciences, vol. 14, no. 22, p. 10086, Nov. 2024, doi: 10.3390/app142210086. Available: https://doi.org/10.3390/app142210086

F. Rashid, B. Doyle, S. C. Han, and S. Seneviratne, "Phishing URL detection generalisation using Unsupervised Domain Adaptation," Computer Networks, vol. 245, p. 110398, Apr. 2024, doi: 10.1016/j.comnet.2024.110398. Available: https://doi.org/10.1016/j.comnet.2024.110398

A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," Journal of King Saud University - Computer and Information Sciences, vol. 35, no. 2, pp. 590–611, Jan. 2023, doi: 10.1016/j.jksuci.2023.01.004. Available: https://doi.org/10.1016/j.jksuci.2023.01.004

C. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shukla, "Applications of deep learning for phishing detection: a systematic literature review," Knowledge and Information Systems, vol. 64, no. 6, pp. 1457–1500, May 2022, doi: 10.1007/s10115-022-01672-x. Available: https://doi.org/10.1007/s10115-022-01672-x

H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, and A. Davtyan, "Phishing URL detection with neural networks: an empirical study," Scientific Reports, vol. 14, no. 1, Oct. 2024, doi: 10.1038/s41598-024-74725-6. Available: https://www.nature.com/articles/s41598-024-74725-6

M. Elsadig et al., "Intelligent deep machine learning cyber phishing URL detection based on BERT features extraction," Electronics, vol. 11, no. 22, p. 3647, Nov. 2022, doi: 10.3390/electronics11223647. Available: https://doi.org/10.3390/electronics11223647

E. Y. Zhang, A. D. Cheok, Z. Pan, J. Cai, and Y. Yan, "From Turing to Transformers: A comprehensive review and tutorial on the evolution and applications of generative transformer models," Sci, vol. 5, no. 4, p. 46, Dec. 2023, doi: 10.3390/sci5040046. Available: https://doi.org/10.3390/sci5040046

M. V. Muntean, "Preprocessing methods for improving phishing URL detection," 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1–4, Sep. 2024, doi: 10.1109/inista62901.2024.10683836. Available: https://doi.org/10.1109/inista62901.2024.10683836